



III Jornada Sage/Python (Vigo, 2012)
Representación con Sage de cuencas de puntos finales inducidas por funciones racionales

Luis Javier Hernández Paricio, Miguel Marañón Grandes y María Teresa Rivas Rodríguez

Universidad de La Rioja

21 de junio de 2012

Índice de contenidos

Semiflujos discretos

Definición

Un **semiflujo discreto** *en un conjunto (espacio topológico) X es una aplicación (continua) $\varphi\colon \mathbb{N} \times X \to X$ tal que:*

(i) $\varphi(0, x) = x$, $\forall x \in X$.

(ii) $\varphi(n, \varphi(m, x)) = \varphi(n + m, x)$, $\forall x \in X$, $\forall n, m \in \mathbb{N}$.

Denotaremos a un semiflujo discreto en X por (X, φ) y, cuando no haya lugar a confusión, usaremos X y $n \cdot x = \varphi(n, x)$ para abreviar.

Semiflujos discretos

Definición

Un **semiflujo discreto** *en un conjunto (espacio topológico) X es una aplicación (continua) $\varphi\colon \mathbb{N} \times X \to X$ tal que:*

(i) $\varphi(0, x) = x, \forall x \in X.$

(ii) $\varphi(n, \varphi(m, x)) = \varphi(n + m, x), \forall x \in X, \forall n, m \in \mathbb{N}.$

Denotaremos a un semiflujo discreto en X por (X, φ) y, cuando no haya lugar a confusión, usaremos X y $n \cdot x = \varphi(n, x)$ para abreviar.
Un semiflujo discreto (X, φ) induce una aplicación (continua) $\varphi^1\colon X \to X$ y una aplicación (continua) $f\colon X \to X$ induce un semiflujo discreto $\varphi\colon \mathbb{N} \times X \to X, \varphi(n, x) = f^n(x).$

Semiflujos discretos

Definición

Un **semiflujo discreto** *en un conjunto (espacio topológico) X es una aplicación (continua) $\varphi\colon \mathbb{N} \times X \to X$ tal que:*

(i) $\varphi(0, x) = x$, $\forall x \in X$.

(ii) $\varphi(n, \varphi(m, x)) = \varphi(n + m, x)$, $\forall x \in X$, $\forall n, m \in \mathbb{N}$.

Denotaremos a un semiflujo discreto en X por (X, φ) y, cuando no haya lugar a confusión, usaremos X y $n \cdot x = \varphi(n, x)$ para abreviar.

Un semiflujo discreto (X, φ) induce una aplicación (continua) $\varphi^1\colon X \to X$ y una aplicación (continua) $f\colon X \to X$ induce un semiflujo discreto $\varphi\colon \mathbb{N} \times X \to X$, $\varphi(n, x) = f^n(x)$.

Dado un semiflujo discreto $X = (X, f)$, la **trayectoria** de un punto $x \in X$, φ_x, viene dada por la sucesión $(f^n(x))_{n\in\mathbb{N}}$.

Puntos finales asociados a un semiflujo discreto

Dado un espacio métrico (X, d) y un semiflujo discreto $\varphi \colon \mathbb{N} \times X \to X$, llamaremos **semiflujo discreto métrico** al triple (X, d, φ).

Definición

Dado un semiflujo discreto métrico $X = (X, d, f)$*, se define el* **espacio de puntos finales** *de* X *como el conjunto cociente*

$$\Pi(X, d) = \frac{\{(f^n(x))_{n\in\mathbb{N}} \mid x \in X\}}{\sim},$$

donde $(f^n(x)) \sim (f^n(y))$ *si y sólo si* $(d(f^n(x), f^n(y))) \to 0$, $x, y \in X$.
Un elemento $a = [(f^n(x))] \in \Pi(X, d)$ *se denomina* **punto final** *del semiflujo discreto métrico.*

Cuencas de puntos finales

Podemos definir la aplicación natural

$$\omega\colon X \to \Pi(X, d),$$

dada por $\omega(x) = [(f^n(x))] = [(x, f(x), f^2(x), \dots)]$.

Cuencas de puntos finales

Podemos definir la aplicación natural

$$\omega\colon X \to \Pi(X, d),$$

dada por $\omega(x) = [(f^n(x))] = [(x, f(x), f^2(x), \dots)]$.

Definición

Sea (X, d) un semiflujo discreto métrico. El subconjunto denotado por $X_a = \omega^{-1}(a)$, $a \in \Pi(X, d)$ se denomina **cuenca del punto final** *a. Existe una partición inducida en X:*

$$X = \bigsqcup_{a \in \Pi(X,d)} X_a,$$

a la que llamaremos **ω-descomposición** *del semiflujo discreto métrico (X, d).*

Puntos fijos y periódicos

Definición

Sea X un semiflujo discreto y x un punto de X.

(i) *x es un* **punto fijo** *si para todo $n \in \mathbb{N}$, $n \cdot x = x$.*

(ii) *x es un* **punto periódico** *si existe $n \in \mathbb{N}$, $n \neq 0$, tal que $n \cdot x = x$.*

Los subconjuntos de los puntos fijos y periódicos de un semiflujo discreto se denotarán por $\mathrm{Fix}(X)$ y $P(X)$, respectivamente.

Puntos fijos y periódicos

Definición

Sea X un semiflujo discreto y x un punto de X.

(i) *x es un* **punto fijo** *si para todo $n \in \mathbb{N}$, $n \cdot x = x$.*

(ii) *x es un* **punto periódico** *si existe $n \in \mathbb{N}$, $n \neq 0$, tal que $n \cdot x = x$.*

Los subconjuntos de los puntos fijos y periódicos de un semiflujo discreto se denotarán por $\mathrm{Fix}(X)$ y $P(X)$, respectivamente.

Sea (X, d) un semiflujo discreto métrico. Si $y \in \mathrm{Fix}(X)$, podemos interpretar que y es un punto final:

$$y \equiv [(y)] = [(y, y, \dots)] \in \Pi(X, d).$$

Estructuras diferenciables y complejas en $\mathbb{C} \cup \{\infty\}$

Se pretende estudiar funciones racionales complejas extendidas sobre la esfera de Riemann $h\colon \mathbb{C} \cup \{\infty\} \to \mathbb{C} \cup \{\infty\}$.

Para ello, se consideran las siguientes estructuras:

$$S^2 \cong \mathbb{C} \cup \{\infty\} \cong \mathbf{P}^1(\mathbb{C})$$

S^2 variedad diferenciable de dimensión 2
$\mathbf{P}^1(\mathbb{C})$ variedad compleja de dimensión 1

Biyección entre $\mathbf{P}^1(\mathbb{C})$ y S^2

Consideremos la biyección $\theta \colon S^2 \to \mathbf{P}^1(\mathbb{C})$ dada por:

$$\theta(r_1, r_2, r_3) = [r_1 + ir_2, 1 - r_3].$$

La aplicación inversa de θ, $\theta^{-1} \colon \mathbf{P}^1(\mathbb{C}) \to S^2$, viene dada por la fórmula:

$$\theta^{-1}([z, t]) = \left(\frac{\bar{z}t + z\bar{t}}{\bar{t}t + z\bar{z}}, \frac{i(\bar{z}t - z\bar{t})}{\bar{t}t + z\bar{z}}, \frac{-\bar{t}t + z\bar{z}}{\bar{t}t + z\bar{z}} \right).$$

Coordenadas homogéneas normalizadas

Dado un punto $[z, t] \in \mathbf{P}^1(\mathbb{C})$, al par (z, t) se le conoce como las coordenadas homogéneas del punto y t/z (z/t) son las coordenadas absolutas de dicho punto.

En los algoritmos, usaremos las siguientes **coordenadas homogéneas normalizadas** para cualquier punto de $\mathbf{P}^1(\mathbb{C})$:

$$[z, t] = \begin{cases} [z/t, 1], & \text{si } |t| \geq |z|, \\ [1, t/z], & \text{si } |t| < |z|. \end{cases}$$

El uso de coordenadas homogéneas normalizadas evitará el desbordamiento de las coordenadas z, t en nuestros programas.

Funciones racionales complejas

Sea $h(u) = \dfrac{a(u^p + a_1 u^{p-1} + \cdots + a_p)}{(u^q + b_1 u^{q-1} + \cdots + b_q)}$ la expresión de una función racional compleja, $a \in \mathbb{C}$, $a \neq 0$. Si tomamos $u = z/t$ y $n = \text{máx}\{p, q\}$, tenemos:

$$\frac{a(z^p t^{n-p} + a_1 z^{p-1} t^{n-p+1} + \cdots + a_p t^n)}{z^q t^{n-q} + b_1 z^{q-1} t^{n-q+1} + \cdots + b_q t^n}.$$

Tanto el numerador $F(z,t) = a(z^p t^{n-p} + a_1 z^{p-1} t^{n-p+1} + \cdots + a_p t^n)$ como el denominador $G(z,t) = z^q t^{n-q} + b_1 z^{q-1} t^{n-q+1} + \cdots + b_q t^n$ son polinomios homogéneos en las variables z, t de grado n, de forma que

$$[F(z,t), G(z,t)] \in \mathbf{P}^1(\mathbb{C}), \quad h(u) = \frac{F(u,1)}{G(u,1)}.$$

Esta técnica permite trabajar con funciones racionales complejas en $\mathbf{P}^1(\mathbb{C})$ utilizando coordenadas homogéneas normalizadas.

Puntos fijos de una función racional

Lema

Si f es una función racional compleja representada por un par de polinomios homogéneos coprimos $A(z,t), B(z,t)$ de grado n, entonces el conjunto $[z_1, t_1], \ldots, [z_{n+1}, t_{n+1}]$ de ceros de $A(z,t)t - B(z,t)z$ es el conjunto de puntos fijos de f.

Puntos fijos de una función racional

Lema

Si f es una función racional compleja representada por un par de polinomios homogéneos coprimos $A(z,t), B(z,t)$ de grado n, entonces el conjunto $[z_1,t_1],\ldots,[z_{n+1},t_{n+1}]$ de ceros de $A(z,t)t - B(z,t)z$ es el conjunto de puntos fijos de f.

Lema

Si f es una función racional compleja representada por un par de polinomios homogéneos coprimos $A(z,t), B(z,t)$ de grado n, entonces f^r es una función racional de grado n^r que tiene n^r+1 puntos fijos (teniéndose en cuenta la multiplicidad).

Métrica cordal en $\mathbf{P}^1(\mathbb{C})$

Como S^2 es un subespacio de $\mathbb{R}^3$, la métrica euclídea usual de $\mathbb{R}^3$ induce una métrica euclídea d^E en S^2.

Usando la biyección $\theta^{-1} \colon \mathbf{P}^1(\mathbb{C}) \to S^2$, podemos trasladar las estructuras métricas de S^2 a $\mathbf{P}^1(\mathbb{C})$ mediante la expresión:

$$d_1^E([z,t],[z',t']) = d^E(\theta^{-1}([z,t]), \theta^{-1}([z',t'])).$$

Métrica cordal en $\mathbf{P}^1(\mathbb{C})$

Como S^2 es un subespacio de $\mathbb{R}^3$, la métrica euclídea usual de $\mathbb{R}^3$ induce una métrica euclídea d^E en S^2.

Usando la biyección $\theta^{-1}\colon \mathbf{P}^1(\mathbb{C}) \to S^2$, podemos trasladar las estructuras métricas de S^2 a $\mathbf{P}^1(\mathbb{C})$ mediante la expresión:

$$d_1^E([z,t],[z',t']) = d^E(\theta^{-1}([z,t]),\theta^{-1}([z',t'])).$$

Una fórmula explícita para d_1^E viene dada por:

$$d_1^E([z,t],[z',t']) =$$

$$= \left(\left(\frac{\bar{z}t + z\bar{t}}{\bar{t}t + z\bar{z}} - \frac{\bar{z'}t' + z\bar{t'}}{\bar{t'}t' + z'\bar{z'}} \right)^2 + \left(\frac{i(\bar{z}t - z\bar{t})}{\bar{t}t + z\bar{z}} - \frac{i(\bar{z'}t' - z'\bar{t'})}{\bar{t'}t' + z'\bar{z'}} \right)^2 + \left(\frac{-\bar{t}t + z\bar{z}}{\bar{t}t + z\bar{z}} - \frac{-\bar{t'}t' + z'\bar{z'}}{\bar{t'}t' + z'\bar{z'}} \right)^2 \right)^{\frac{1}{2}}.$$

Aplicación tangente de una función racional

Definición

Sea $f : \mathbf{P}^1(\mathbb{C}) \to \mathbf{P}^1(\mathbb{C})$ *una función analítica y* $p \in \mathbf{P}^1(\mathbb{C})$ *un punto fijo. Entonces, se dice que* p *es* **superatractor**, **atractor**, **indiferente** *o* **repulsor** *si la norma de la aplicación tangente en ese punto es* 0, *menor que* 1, *igual a* 1 *ó mayor que* 1, *respectivamente.*

Aplicación tangente de una función racional

Definición

Sea $f\colon \mathbf{P}^1(\mathbb{C}) \to \mathbf{P}^1(\mathbb{C})$ una función analítica y $p \in \mathbf{P}^1(\mathbb{C})$ un punto fijo. Entonces, se dice que p es **superatractor**, **atractor**, **indiferente** *o* **repulsor** *si la norma de la aplicación tangente en ese punto es 0, menor que 1, igual a 1 ó mayor que 1, respectivamente.*

Sean $x, y\colon \mathbf{P}^1(\mathbb{C}) \to S^2$ cartas dadas por $x([z,t]) = z/t$ e $y([z,t]) = t/z$, $\mathrm{Dom}\, x = \{[z,t] \in \mathbf{P}^1(\mathbb{C}) \mid t \neq 0\}$, $\mathrm{Dom}\, y = \{[z,t] \in \mathbf{P}^1(\mathbb{C}) \mid z \neq 0\}$.
Dado $p = [z,t] \in \mathbf{P}^1(\mathbb{C})$, para determinar si un punto fijo es superatractor, atractor, indiferente o repulsor basta comprobar si

$$|J_p(f)| = \begin{cases} \text{Abs}(xfx^{-1})'(z), & \text{si t=1, t'=1,} \\ \text{Abs}(yfy^{-1})'(t), & \text{si z=1, z'=1} \end{cases}$$

es 0, menor que 1, igual a 1 ó mayor que 1.

Semiflujos discretos inducidos por una función racional

Cualquier función racional compleja $h\colon \mathbb{C} \to \mathbb{C}$ induce una aplicación extensión $f = h^+\colon \mathbb{C} \cup \{\infty\} \to \mathbb{C} \cup \{\infty\}$ que dota a $\mathbb{C} \cup \{\infty\}$ de una estructura de semiflujo discreto mediante la fórmula $n \cdot p = f^n(p)$.

Semiflujos discretos inducidos por una función racional

Cualquier función racional compleja $h\colon \mathbb{C} \to \mathbb{C}$ induce una aplicación extensión $f = h^{+}\colon \mathbb{C} \cup \{\infty\} \to \mathbb{C} \cup \{\infty\}$ que dota a $\mathbb{C} \cup \{\infty\}$ de una estructura de semiflujo discreto mediante la fórmula $n \cdot p = f^{n}(p)$.

Por otro lado, disponemos de la aplicación natural

$$\omega\colon \mathbb{C} \cup \{\infty\} \to \Pi(\mathbb{C} \cup \{\infty\})$$

dada por $\omega(p) = [(p, f(p), f^2(p), \dots)]$, la cual permitirá descomponer el espacio $\mathbb{C} \cup \{\infty\}$ en cuencas de atracción disjuntas asociadas a puntos finales.

Ejemplo

En el siguiente ejemplo estudiamos la función racional $h(z) = \frac{F(z)}{G(z)}$, donde $F(z) = 4z^5 + 1$ y $G(z) = 5z^4$. En este caso, la aplicación inducida $f = h^+$ en $X = \mathbb{C} \cup \{\infty\}$ posee seis puntos fijos:

$$p_0 = \infty, \quad p_1 = -0{,}809017 - 0{,}587785i, \quad p_2 = -0{,}809017 + 0{,}587785i,$$

$$p_3 = 0{,}309017 - 0{,}951057i, \quad p_4 = 0{,}309017 + 0{,}951057i, \quad p_5 = 1.$$

Ejemplo

En el siguiente ejemplo estudiamos la función racional $h(z) = \frac{F(z)}{G(z)}$, donde $F(z) = 4z^5 + 1$ y $G(z) = 5z^4$. En este caso, la aplicación inducida $f = h^+$ en $X = \mathbb{C} \cup \{\infty\}$ posee seis puntos fijos:

$$p_0 = \infty, \quad p_1 = -0{,}809017 - 0{,}587785i, \quad p_2 = -0{,}809017 + 0{,}587785i,$$

$$p_3 = 0{,}309017 - 0{,}951057i, \quad p_4 = 0{,}309017 + 0{,}951057i, \quad p_5 = 1.$$

Por tanto, consideraremos la esfera dividida en siete partes:

$$X = (X \setminus D) \sqcup D_\infty \sqcup D_{p_1} \sqcup D_{p_2} \sqcup D_{p_3} \sqcup D_{p_4} \sqcup D_{p_5},$$

donde $D = D_\infty \cup D_{p_1} \cup D_{p_2} \cup D_{p_3} \cup D_{p_4} \cup D_{p_5}$.

Relación entre regiones, colores y puntos fijos

A cada región se le asocia un color diferente:

Región	Color	Tipo de punto fijo asociado
$X \setminus D$	0	
$D_{p_0} = \omega^{-1}\omega(p_0)$	1	Repulsor
$D_{p_1} = \omega^{-1}\omega(p_1)$	2	Atractor
$D_{p_2} = \omega^{-1}\omega(p_2)$	3	Atractor
$D_{p_3} = \omega^{-1}\omega(p_3)$	4	Atractor
$D_{p_4} = \omega^{-1}\omega(p_4)$	5	Atractor
$D_{p_5} = \omega^{-1}\omega(p_5)$	6	Atractor

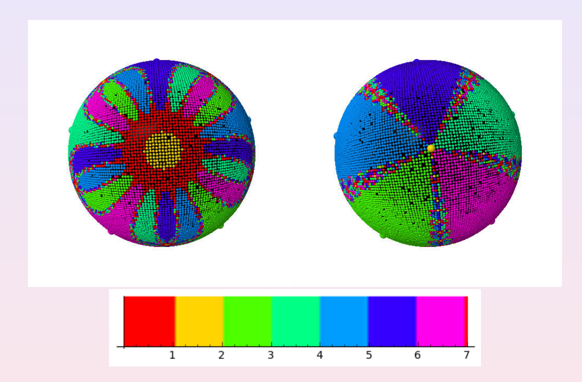
Relación entre regiones, colores y puntos fijos

A cada región se le asocia un color diferente:

Región	Color	Tipo de punto fijo asociado
$X \setminus D$	0	
$D_{p_0} = \omega^{-1}\omega(p_0)$	1	Repulsor
$D_{p_1} = \omega^{-1}\omega(p_1)$	2	Atractor
$D_{p_2} = \omega^{-1}\omega(p_2)$	3	Atractor
$D_{p_3} = \omega^{-1}\omega(p_3)$	4	Atractor
$D_{p_4} = \omega^{-1}\omega(p_4)$	5	Atractor
$D_{p_5} = \omega^{-1}\omega(p_5)$	6	Atractor

En $X \setminus D$ se pueden encontrar puntos cuya cuenca de atracción está asociada a un punto final que no está vinculado a ningún punto fijo (por ejemplo, cuencas de 2-ciclos, 3-ciclos,...) o puntos tales que, tras un número prefijado de iteraciones, forman parte de una sucesión que aún no ha convergido a ningún punto fijo. Los demás colores corresponden a las cuencas asociadas a los diferentes puntos fijos.

Representación de las cuencas de atracción



La esfera de la izquierda muestra las regiones cercanas al origen (polo sur de la esfera), mientras que la de la derecha muestra las próximas al punto del infinito (polo norte de la esfera).

Cálculo de los puntos fijos

Hemos visto que el conjunto $[z_1, t_1], \ldots, [z_{n+1}, t_{n+1}]$ de ceros del polinomio homogéneo $A(z,t)t - B(z,t)z$ coincide con el conjunto de puntos fijos de $f(z,t) = \dfrac{A(z,t)}{B(z,t)}$.

Cálculo de los puntos fijos

Hemos visto que el conjunto $[z_1, t_1], \ldots, [z_{n+1}, t_{n+1}]$ de ceros del polinomio homogéneo $A(z,t)t - B(z,t)z$ coincide con el conjunto de puntos fijos de $f(z,t) = \dfrac{A(z,t)}{B(z,t)}$.

- Si $t = 0$ es raíz de $B(1,t)$ y $z_1, \ldots, z_n$ son las raíces de $A(z,1) - B(z,1)z = 0$, entonces $\{[1,0],[z_1,1],\ldots,[z_n,1]\}$ es el conjunto de los puntos fijos de f.
- Si $t = 0$ no es raíz de $B(1,t)$ y $z_1, \ldots, z_n, z_{n+1}$ son las raíces de $A(z,1) - B(z,1)z = 0$, entonces $\{[z_1,1],\ldots,[z_n,1],[z_{n+1},1]\}$ es el conjunto de los puntos fijos de f.

fixedPointsZeros

```
def fixedPointsZeros (U, V):
    w = var('w')
    con = V(z = 1, t = 0)
    solaux = solve(U(z = w, t = 1)-(V(z = w, t = 1))*w == 0, w,
        to_poly_solve = true, solution_dict = true)
    def errorEq():
        boo = false
        k = 0
        while k < len(solaux) and boo == false:
            if w in solaux[k].keys(): k = k + 1
            else: boo = true
        return boo
    if errorEq():
        sol1=[]
        print "There was a problem with function solve:
            cannot solve equation", U(w, 1)-(V(w, 1))*w == 0
    else:
        sol1=[homogeneousNormalization((n(t[w]),1)) for t in solaux]
        if (con == 0):
            sol1.insert(0, (1, 0))
    return sol1
```

Distancia entre dos puntos

Utilizando la biyección dada entre $\mathbf{P}^1(\mathbb{C})$ y S^2 y la métrica euclídea en S^2, podemos obtener la distancia entre dos puntos de $\mathbf{P}^1(\mathbb{C})$.

```
sphereBijection((1,0))
(0,0,1)
chordalMetric((1,0),(1,1))
1.41421356237310
```

sphereBijection

```
def sphereBijection(twotuple):
    z = twotuple[0]
    t = twotuple[1]
    return ((conjugate(z)*t + conjugate(t)*z) /
            (conjugate(t)*t + conjugate(z)*z),
        (I*(conjugate(z)*t - conjugate(t)*z)) /
            (conjugate(t)*t + conjugate(z)*z),
        (-conjugate(t)*t + conjugate(z)*z) /
            (conjugate(t)*t + conjugate(z)*z))
```

chordalMetric

```
def chordalMetric(twotuple, twotuple1):
    t1 = sphereBijection(twotuple)
    t2 = sphereBijection(twotuple1)
    m1 = Matrix([[t1[0], t1[1], t1[2]]]);
    m2 = Matrix([[t2[0], t2[1], t2[2]]]);
    return n(norm(m1-m2))
```

Iteración de la función racional

Con el fin de encontrar el punto final asociado a un punto $x \in \mathbf{P}^1(\mathbb{C})$, se tiene que iterar (componer consigo misma) la función f para obtener una sucesión finita $(x, f(x), f^2(x), f^3(x), \ldots, f^{k-1}(x), f^k(x))$. Al programar en el ordenador, se debe considerar un número máximo de iteraciones l y prefijar una precisión c para determinar cuándo debemos detener el proceso iterativo. Es por ello que trabajaremos siempre con sucesiones tales que $k \leq l$.

Iteración de la función racional

Con el fin de encontrar el punto final asociado a un punto $x \in \mathbf{P}^1(\mathbb{C})$, se tiene que iterar (componer consigo misma) la función f para obtener una sucesión finita $(x, f(x), f^2(x), f^3(x), \ldots, f^{k-1}(x), f^k(x))$. Al programar en el ordenador, se debe considerar un número máximo de iteraciones l y prefijar una precisión c para determinar cuándo debemos detener el proceso iterativo. Es por ello que trabajaremos siempre con sucesiones tales que $k \leq l$.

Después de cada iteración, se nos presentarán dos posibles opciones:

1) Si la distancia cordal desde $f^{k-1}(x)$ hasta $f^k(x)$ es menor que 10^{-c}, entonces tomamos como salida la lista $[f^k(x), k]$; en otro caso, se aplica 2).
2) Si $k < l$, se lleva a cabo una nueva iteración y se aplica de nuevo 1); en otro caso (si $k = l$), se toma como salida $[f^l(x), l]$.

newstep

```
def newstep(U, V, iter, precision, pointinternumber):
    point = pointinternumber
    number = 0
    imagepoint = rationalFunction(U, V, point)
    while (chordalMetric(point, imagepoint) >
            10.**(-precision)) and (number < iter):
        point = imagepoint
        imagepoint = rationalFunction(U, V, point)
        number = number + 1
    else: return [imagepoint, number]
```

Determinación del punto fijo al que converge una trayectoria y del número de iteraciones

Consideremos la sucesión ordenada de puntos fijos $\{x_1, x_2, \ldots, x_{n+1}\}$. Del mismo modo, dado un punto $x \in \mathbb{C} \cup \{\infty\}$, consideremos la sucesión de iteraciones $(x, f(x), f^2(x), f^3(x), \ldots, f^{k-1}(x), f^k(x))$.

Si existe $i \in \{1 \ldots, n+1\}$ tal que la distancia cordal entre $f^k(x)$ y el punto fijo x_i es menor que 10^{-c}, entonces `positionIterationNumber` debe devolver $[i, k]$; en otro caso, $k = l$ y la salida debe ser $[0, l]$, donde l es el número máximo de iteraciones prefijada de antemano.

```
positionIterationNumber(A,B,fixedPointsZeros(A,B),25,4,(-0.1-0.1*i,1))
[0,25]
positionIterationNumber(A,B,fixedPointsZeros(A,B),25,4,(-0.1-0.09*i,1))
[5,23]
```

positionIterationNumber

```
def positionIterationNumber(U, V, fixedpointlist, iter,
        precision, twotuple):
    result = newstep(U, V, iter, precision, twotuple)
    if (result[1] != iter):
        return [position(fixedpointlist, precision,
            result[0]), result[1]]
    else:
        return [0, result[1]]
```

Derivada de una función racional en un punto fijo

Para saber si un punto fijo es superatractor, atractor, indiferente o repulsor, podemos calcular la derivada de la función racional en ese punto usando el siguiente algoritmo:

```
fixedPointsTangentMapNorm(A,B,(1,0))
((1,0),1.33333333333333)
```

Supongamos que $A(z,t), B(z,t)$ son polinomios homogéneos y que $[z,t]$ es un punto fijo representado en forma de coordenadas homogéneas normalizadas. Entonces, el subprograma `fixedPointsTangentMapNorm` devuelve una lista con dos elementos: el punto fijo considerado $[z,t]$ y la norma de la derivada de la función racional en ese punto.

fixedPointsTangentMapNorm

```
a, b = var('a, b')
def fixedPointsTangentMapNorm(A, B, twotuple):
    nor = homogeneousNormalization(twotuple)
    if (nor[1] == 1):
        return (nor,
            abs(derivative(A(a, 1)/B(a, 1),a)(a = nor[0])))
    else:
        return (nor,
            abs(derivative(B(1, b)/A(1, b),b)(b = nor[1])))
```

Cuencas de *n*-ciclos de una función racional



Cuencas de *n*-ciclos de una función racional

- El programa posee un argumento opcional que permite trabajar con las parejas de polinomios asociadas a $f^2, f^3, \ldots$.

Cuencas de n-ciclos de una función racional

- El programa posee un argumento opcional que permite trabajar con las parejas de polinomios asociadas a $f^2, f^3, \ldots$.
- La utilidad de esto radica en que las cuencas de atracción de los puntos fijos de la función f compuesta dos veces consigo misma, $f^2 = f \circ f$, corresponden a cuencas de puntos fijos y de 2-ciclos de f; si dos puntos fijos de f^2 conforman un 2-ciclo, la cuenca de atracción de este 2-ciclo es la unión de las cuencas de estos dos puntos fijos.

Cuencas de *n*-ciclos de una función racional

- El programa posee un argumento opcional que permite trabajar con las parejas de polinomios asociadas a $f^2, f^3, \ldots$.
- La utilidad de esto radica en que las cuencas de atracción de los puntos fijos de la función f compuesta dos veces consigo misma, $f^2 = f \circ f$, corresponden a cuencas de puntos fijos y de 2-ciclos de f; si dos puntos fijos de f^2 conforman un 2-ciclo, la cuenca de atracción de este 2-ciclo es la unión de las cuencas de estos dos puntos fijos.
- Este hecho puede generalizarse a una función compuesta n veces, f^n.

compose

```
def compose(h, num):
    H = h
    if (num < 1):
        print("Rational function must be composed once
            at least.")
    for cont in range(num-1): H = h.subs(x = H)
    return H
```

homogenize

```
def homogenize(f, g):
    F = R(0); G = R(0)
    fdeg = f.degree(x)
    gdeg = g.degree(x)
    deg = max(fdeg, gdeg)
    if(fdeg != 0):
        monf = [z^q[1]*q[0] for q in f.coefficients(x)]
        for m in monf:
            d = m.degree(z)
            F = F + m * t^(deg - d)
    else: F = f * t^deg
    if(gdeg != 0):
        mong = [z^q[1]*q[0] for q in g.coefficients(x)]
        for m in mong:
            d = m.degree(z)
            G = G + m * t^(deg - d)
    else: G = g * t^deg
    return [F, G]
```

composeHomogenize

```
def composeHomogenize(f, g, n):
    if(g.degree(x) == 0):
        comp = compose(f, n)
        homo = homogenize(comp, g)
    else:
        h = f / g
        comp = compose(h, n).simplify_rational('simple')
        num = comp.numerator()
        den = comp.denominator()
        homo = homogenize(num, den)
    A = homo[0]
    B = homo[1]
    return [A, B]
```

Estrategias de asignación de color

Dibujaremos un fractal usando el punto fijo al que converge la trayectoria (quizá no exista ninguno) y el número de iteraciones necesarias para alcanzar la convergencia, y una de las siguientes estrategias:

Estrategias de asignación de color

Dibujaremos un fractal usando el punto fijo al que converge la trayectoria (quizá no exista ninguno) y el número de iteraciones necesarias para alcanzar la convergencia, y una de las siguientes estrategias:

1) **Punto fijo al que converge la trayectoria:** se asigna un color a la cuenca de atracción de cada punto fijo, de modo que cada punto x se colorea según el punto fijo al cual $f^k(x)$ converge; se marca con un nuevo color si, después de un cierto número de iteraciones, la sucesión no converge.

Estrategias de asignación de color

Dibujaremos un fractal usando el punto fijo al que converge la trayectoria (quizá no exista ninguno) y el número de iteraciones necesarias para alcanzar la convergencia, y una de las siguientes estrategias:

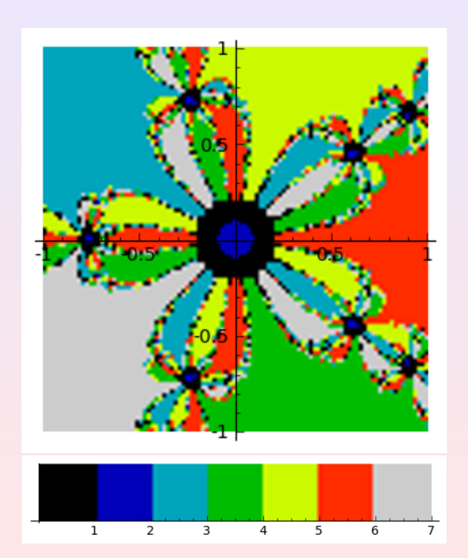
1) **Punto fijo al que converge la trayectoria:** se asigna un color a la cuenca de atracción de cada punto fijo, de modo que cada punto x se colorea según el punto fijo al cual $f^k(x)$ converge; se marca con un nuevo color si, después de un cierto número de iteraciones, la sucesión no converge.
2) **Número de iteraciones:** se asigna el color atendiendo al número de iteraciones necesarias para alcanzar algún punto fijo con una determinada precisión.

Estrategias de asignación de color

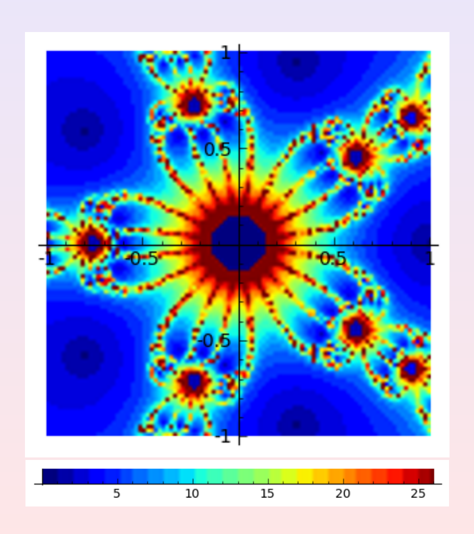
Dibujaremos un fractal usando el punto fijo al que converge la trayectoria (quizá no exista ninguno) y el número de iteraciones necesarias para alcanzar la convergencia, y una de las siguientes estrategias:

1) **Punto fijo al que converge la trayectoria:** se asigna un color a la cuenca de atracción de cada punto fijo, de modo que cada punto x se colorea según el punto fijo al cual $f^k(x)$ converge; se marca con un nuevo color si, después de un cierto número de iteraciones, la sucesión no converge.
2) **Número de iteraciones:** se asigna el color atendiendo al número de iteraciones necesarias para alcanzar algún punto fijo con una determinada precisión.
3) **Combinación de las estrategias anteriores:** se asigna un color a cada cuenca de atracción de un punto fijo, pero haciendo que este color sea más claro o más oscuro en función del número de iteraciones necesarias para alcanzar la raíz con la precisión prefijada.

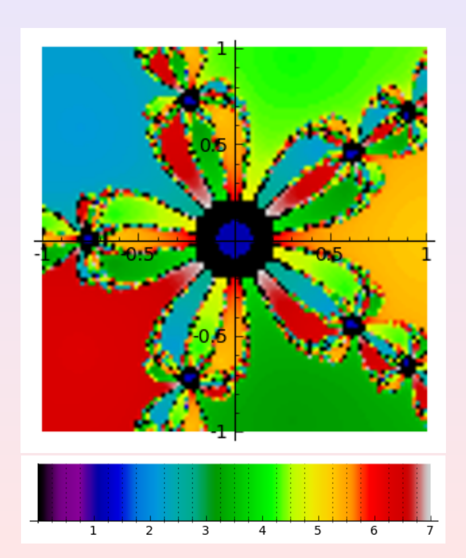
Punto fijo al que converge la trayectoria



Número de iteraciones



Combinación de las estrategias anteriores



Algoritmos

Algoritmos

- Con la función `fractalPlot` obtenemos un fractal coloreado en un **rectángulo**.

Algoritmos

- Con la función `fractalPlot` obtenemos un fractal coloreado en un **rectángulo**.
- `fractalPlotInsideOutside` devuelve **dos discos**: en el primero se representa la intersección de las cuencas de atracción con el **disco unidad** y en el segundo, mediante **inversión**, obtenemos la intersección de las cuencas con el complementario del disco en $\mathbb{C} \cup \{\infty\}$.

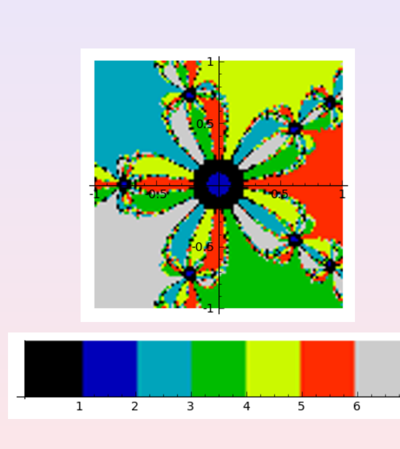
Algoritmos

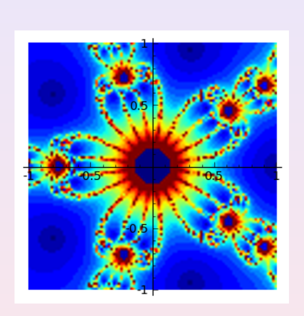
- Con la función `fractalPlot` obtenemos un fractal coloreado en un **rectángulo**.
- `fractalPlotInsideOutside` devuelve **dos discos**: en el primero se representa la intersección de las cuencas de atracción con el **disco unidad** y en el segundo, mediante **inversión**, obtenemos la intersección de las cuencas con el complementario del disco en $\mathbb{C} \cup \{\infty\}$.
- Con la función `spherePlot` se obtiene un **fractal 3D en la esfera unidad** en el que se muestran todos los puntos fijos con un tamaño mayor que el resto de puntos.

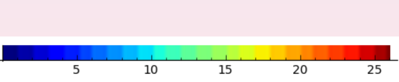
Algoritmos

- Con la función `fractalPlot` obtenemos un fractal coloreado en un **rectángulo**.
- `fractalPlotInsideOutside` devuelve **dos discos**: en el primero se representa la intersección de las cuencas de atracción con el **disco unidad** y en el segundo, mediante **inversión**, obtenemos la intersección de las cuencas con el complementario del disco en $\mathbb{C} \cup \{\infty\}$.
- Con la función `spherePlot` se obtiene un **fractal 3D en la esfera unidad** en el que se muestran todos los puntos fijos con un tamaño mayor que el resto de puntos.
- El subprograma `cubicSpherePlot` devuelve lo mismo que `spherePlot`, pero la esfera que se obtiene con la primera función es ligeramente distinta a la que se obtiene con la segunda, pues sus puntos se distribuyen sobre su superficie de una forma diferente (concretamente, **proyectando un cubo** sobre la esfera unidad).

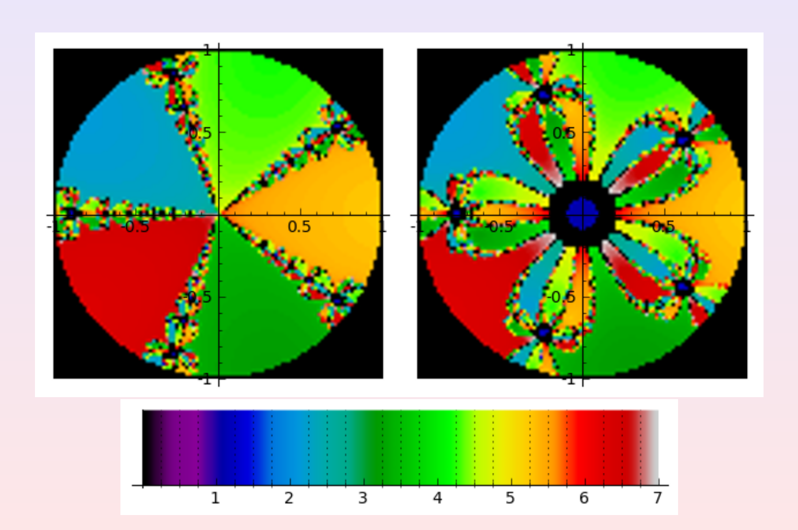
fractalPlot



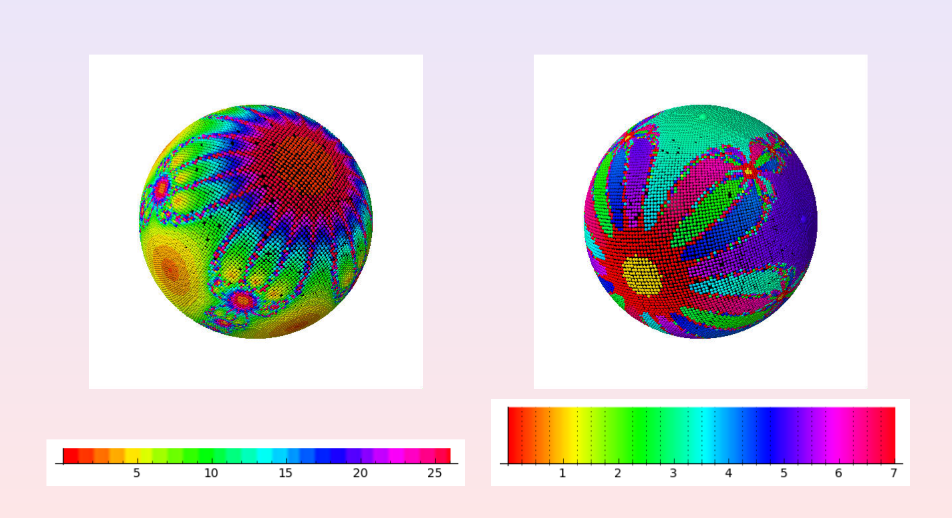




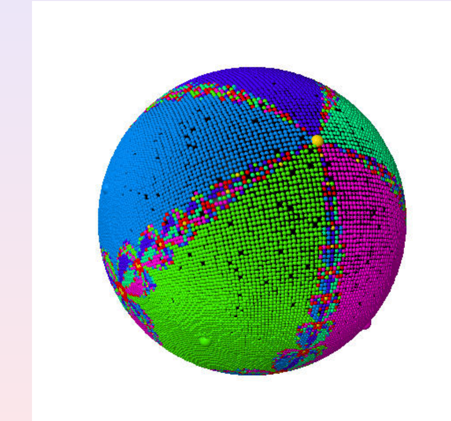
fractalPlotInsideOutside

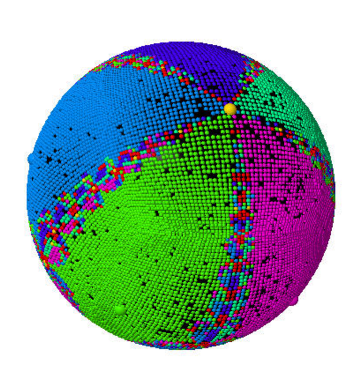


spherePlot



spherePlot vs cubicSpherePlot





Ejemplo de uso

El programa es muy fácil de usar. Para dibujar el fractal asociado a una determinada función racional, es suficiente con especificar los polinomios que forman su numerador y denominador en la variable x y ejecutar una de las siguientes subrutinas, dependiendo del tipo de fractal que queramos dibujar: `fractalPlotInsideOutside`, `fractalPlot`, `spherePlot` o `cubicSpherePlot`.

```
M=4*x**5+1; N=5*x**4
fractalPlotInsideOutside(M,N,100,positionPlusConvergence)
```

fractalPlot

- **fractalPlot**(*M, N, xmin, xmax, ymin, ymax, points = 100, function = onlyPosition, iter = 25, prec = 3, ncomp = 1, colorfunction = 'spectral'*)
  - `M,N` son el numerador y el denominador de la función racional dada en la variable x, respectivamente.
  - La tupla `xmin,xmax,ymin,ymax` representa los vértices del rectángulo en el que se dibujará el fractal.
  - `points` es un entero que representa el número de puntos del gráfico.
  - `function` indica la estrategia que se empleará al dibujar el fractal.
  - `iter` es un entero que representa el número máximo de iteraciones.
  - `prec` es un entero tal que, si la distancia entre dos puntos es menor que $10^{-\text{prec}}$, entonces el algoritmo considera que los dos puntos son el mismo.
  - `ncomp` es un entero que representa el número de veces que la función racional se compone consigo misma.
  - `colorfunction` es un colormap de Sage que se usa para asignar un color a cada punto del plano complejo.

`fractalPlotInsideOutside`

- **fractalPlotInsideOutside**(*M, N, points = 100, function = onlyPosition, iter = 25, prec = 3, ncomp = 1, reflection = -1, colorfunction = 'spectral'*)
  - `M,N` son el numerador y el denominador de la función racional dada en la variable x, respectivamente.
  - `points` es un entero que representa el número de puntos del gráfico.
  - `function` indica la estrategia que se empleará al dibujar el fractal.
  - `iter` es un entero que representa el número máximo de iteraciones.
  - `prec` es un entero tal que, si la distancia entre dos puntos es menor que $10^{-\mathrm{prec}}$, entonces el algoritmo considera que los dos puntos son el mismo.
  - `ncomp` es un entero que representa el número de veces que la función racional se compone consigo misma.
  - `reflection` es un número igual a 1 ó -1 que indica el signo de la reflexión del método de inversión.
  - `colorfunction` es un colormap de Sage que se usa para asignar un color a cada punto del plano complejo.

`spherePlot`

- **spherePlot**(*M, N, points = 100, function = onlyPosition, iter = 25, prec = 3, ncomp = 1*)
  - `M,N` son el numerador y el denominador de la función racional dada en la variable x, respectivamente.
  - `points` es un entero que representa el número de puntos del gráfico.
  - `function` indica la estrategia que se empleará al dibujar el fractal.
  - `iter` es un entero que representa el número máximo de iteraciones.
  - `prec` es un entero tal que, si la distancia entre dos puntos es menor que $10^{-\text{prec}}$, entonces el algoritmo considera que los dos puntos son el mismo.
  - `ncomp` es un entero que representa el número de veces que la función racional se compone consigo misma.

cubicSpherePlot

- **cubicSpherePlot**(*M, N, numdiv = 60, function = onlyPosition, iter = 25, prec = 3, ncomp = 1*)
  - `M,N` son el numerador y el denominador de la función racional dada en la variable x, respectivamente.
  - `numdiv` es un entero que indica el número de subdivisiones de las caras del cubo que se proyecta sobre la esfera unidad.
  - `function` indica la estrategia que se empleará al dibujar el fractal.
  - `iter` es un entero que representa el número máximo de iteraciones.
  - `prec` es un entero tal que, si la distancia entre dos puntos es menor que $10^{-\mathrm{prec}}$, entonces el algoritmo considera que los dos puntos son el mismo.
  - `ncomp` es un entero que representa el número de veces que la función racional se compone consigo misma.

Funciones racionales inducidas por métodos iterativos numéricos

- Todos los ceros de un polinomio complejo son puntos fijos de la función racional obtenida a partir de él mediante los métodos numéricos iterativos más usuales.
- Cada punto fijo de esta función racional asociada puede interpretarse directamente como un punto final.
- Así, toda cuenca de atracción de una raíz del polinomio complejo primitivo es precisamente la cuenca del punto final asociado.
- Si p es una raíz del polinomio complejo original, el significado de que un punto x esté en la cuenca correspondiente al punto fijo p es que, al aplicar a ese punto x la función racional asociada al método numérico con el que estamos trabajando, estaremos aproximándonos cada vez más a la raíz p del polinomio dado.

Conexiones con la Geometría Fractal

Intuitivamente, se puede decir que un punto $x \in X$ pertenece al **conjunto de Fatou** si existe un entorno abierto U de x tal que $\omega(x) = \omega(y)$, $\forall y \in U$ (es decir, si la cuenca de un punto final asociada a cualquier punto muy cercano a x es la misma que la cuenca del punto final a la que pertenece x).

El **conjunto de Julia** es la frontera de las cuencas de atracción de los puntos fijos atractores de f, incluyendo a ∞. En consecuencia, se puede considerar que el conjunto de Julia está formado por puntos que se encuentran en la frontera de las cuencas de puntos atractores (y por tanto, el resto de puntos de X están en el conjunto de Fatou).

Aplicaciones futuras



Aplicaciones futuras

- Nuestros algoritmos pueden emplearse con el propósito de obtener una estimación numérica del **área en la esfera de las cuencas de atracción** asociadas a las raíces de un polinomio, así como la **probabilidad** de que un punto $x_0 \in \mathbb{C} \cup \{\infty\}$ pertenezca a una u otra región.

Aplicaciones futuras

- Nuestros algoritmos pueden emplearse con el propósito de obtener una estimación numérica del **área en la esfera de las cuencas de atracción** asociadas a las raíces de un polinomio, así como la **probabilidad** de que un punto $x_0 \in \mathbb{C} \cup \{\infty\}$ pertenezca a una u otra región.
- Algunas partes de los algoritmos pueden ser también útiles para hacer un estudio más detallado de los conjuntos de Julia de los fractales obtenidos, calculando, por ejemplo, su **dimensión fractal** y otros **invariantes homotópicos y homológicos**.

Aplicaciones futuras

- Nuestros algoritmos pueden emplearse con el propósito de obtener una estimación numérica del **área en la esfera de las cuencas de atracción** asociadas a las raíces de un polinomio, así como la **probabilidad** de que un punto $x_0 \in \mathbb{C} \cup \{\infty\}$ pertenezca a una u otra región.
- Algunas partes de los algoritmos pueden ser también útiles para hacer un estudio más detallado de los conjuntos de Julia de los fractales obtenidos, calculando, por ejemplo, su **dimensión fractal** y otros **invariantes homotópicos y homológicos**.
- Se ha comenzado a desarrollar **en Sage** un algoritmo que permite hallar los **números de Betti** asociados a las cuencas de atracción de puntos finales inducidas por funciones racionales mediante el cálculo de la **homología de complejos cúbicos**.

Dificultades encontradas al programar en Sage



Dificultades encontradas al programar en Sage

- Incoherencias entre los atributos `degree`, `coefficients`, `monomials` de los polinomios, dependiendo de si son de una o de varias variables y del anillo en el que se definen.

Dificultades encontradas al programar en Sage

- Incoherencias entre los atributos `degree`, `coefficients`, `monomials` de los polinomios, dependiendo de si son de una o de varias variables y del anillo en el que se definen.
- Manejo de constantes en determinados anillos.

Dificultades encontradas al programar en Sage

- Incoherencias entre los atributos `degree`, `coefficients`, `monomials` de los polinomios, dependiendo de si son de una o de varias variables y del anillo en el que se definen.
- Manejo de constantes en determinados anillos.
- Falta de correspondencia entre las paletas de color asociadas a gráficos en dos y tres dimensiones.

Dificultades encontradas al programar en Sage

- Incoherencias entre los atributos `degree`, `coefficients`, `monomials` de los polinomios, dependiendo de si son de una o de varias variables y del anillo en el que se definen.
- Manejo de constantes en determinados anillos.
- Falta de correspondencia entre las paletas de color asociadas a gráficos en dos y tres dimensiones.
- "Reorganización" automática indebida de los colores de una paleta de color en los gráficos obtenidos con `density_plot`.

Dificultades encontradas al programar en Sage

- Incoherencias entre los atributos `degree`, `coefficients`, `monomials` de los polinomios, dependiendo de si son de una o de varias variables y del anillo en el que se definen.
- Manejo de constantes en determinados anillos.
- Falta de correspondencia entre las paletas de color asociadas a gráficos en dos y tres dimensiones.
- "Reorganización" automática indebida de los colores de una paleta de color en los gráficos obtenidos con `density_plot`.
- Jmol: problemas al cargar los gráficos y puntos que no aparecen al dibujar las esferas. *¿ParametricPlot3D de Mathematica?*

Dificultades encontradas al programar en Sage

- Incoherencias entre los atributos `degree`, `coefficients`, `monomials` de los polinomios, dependiendo de si son de una o de varias variables y del anillo en el que se definen.
- Manejo de constantes en determinados anillos.
- Falta de correspondencia entre las paletas de color asociadas a gráficos en dos y tres dimensiones.
- "Reorganización" automática indebida de los colores de una paleta de color en los gráficos obtenidos con `density_plot`.
- Jmol: problemas al cargar los gráficos y puntos que no aparecen al dibujar las esferas. *¿ParametricPlot3D de Mathematica?*
- Lentitud en los cálculos.

Dificultades encontradas al programar en Sage

- Incoherencias entre los atributos `degree`, `coefficients`, `monomials` de los polinomios, dependiendo de si son de una o de varias variables y del anillo en el que se definen.
- Manejo de constantes en determinados anillos.
- Falta de correspondencia entre las paletas de color asociadas a gráficos en dos y tres dimensiones.
- "Reorganización" automática indebida de los colores de una paleta de color en los gráficos obtenidos con `density_plot`.
- Jmol: problemas al cargar los gráficos y puntos que no aparecen al dibujar las esferas. *¿ParametricPlot3D de Mathematica?*
- Lentitud en los cálculos.
- *Y por supuesto... poca experiencia programando en Sage.*

Muchas gracias por su atención